UNITED STATES PATENT AND TRADEMARK OFFICE

A

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/691,414	10/22/2003	Andrew Nuss	11103	6165

27015 7590 10/20/2005

CHARLES LOUIS THOEMING
1390 WILLOW PASS ROAD, SUITE 1020
CONCORD, CA 94520

EXAMINER	
FOWLKES, ANDRE R	
ART UNIT	PAPER NUMBER
2192	

DATE MAILED: 10/20/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/691,414	NUSS, ANDREW	
	Examiner	Art Unit	
	Andre R. Fowlkes	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _06 July 2005_.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _58-83_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) _58-83_ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed 7/6/05.

Specification

2. The objections to the specification are withdrawn, in view of applicant's amendment.

Claim Objections

3. The objection to claims 44 and 47 is withdrawn, in view of applicant's amendment.

Claim Rejections - 35 USC § 101

4. The rejection of claims 1-57 under 35 U.S.C. 101 is withdrawn, in view of applicant's amendment.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6: Claims 58-83 are rejected under 35 U.S.C. 102(b) as being anticipated by Friedl, “Mastering regular expressions”, 2nd edition, ISBN: 0-596-00289-0.

As per claim 58, Friedl discloses **a computer-implemented method of integrating side-effects into regular expressions,** (p. 4:14-17, ” Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data”)**, the method comprising the steps of:**

- defining a built-in datatype, to represent any regular expressions, or regex composition, in such a way that a regular expression referenced by a variable of such datatype is immutable, and wherein the datatype facilitates composition using variables, and allows enforcement of strong type compatibility for regular expressions, such as when and where regular expressions can be used in the overall grammar of a programming language (p. 33:1-4, ” many programmers independently developed Java regex packages of varying degrees of quality, functionality, and complexity. With the early-2002 release of Java 1.4, Sun entered the fray with their java.util.regex package”),

- defining a grammar for standard regular expression compositions, such as character- classes, string and character literals, a wildcard, repeats, iterates, concatenations, and unions', defining a grammar for general regular expression

compositions that contribute side- effects', defining a grammar for expressing the particular side-effect of capture (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for <u>operator overloading</u>"),

- **defining a grammar for regex compositions in which side-effects are inhibited; defining a grammar for creating named, re-usable encapsulations for regular expressions, wherein the encapsulations also allow parameterization and the integration of side-effects**(p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading"),

- **and defining a grammar for a binary composition, called the subjunctive", including a primary and secondary term, in which the secondary term is used to narrow the specificity of the primary (by qualifying its matches), while allowing the side- effects of the primary to trigger in the expected way** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading"),

- **wherein side-effects are encoded as general statements within the same language or grammar in which the regular expressions are composed** (p. 33:1-4, " many programmers independently developed Java regex packages of varying degrees of quality, functionality, and complexity. With the early-2002 release of Java 1.4, Sun entered the fray with their java.util.regex package"),

- **wherein the side-effects are executed by the same machine or virtual machine that executes all of the translated instructions of the language hosting**

the regular expression (p. 33:1-4, " many programmers independently developed Java regex packages of varying degrees of quality, functionality, and complexity. With the early-2002 release of Java 1.4, Sun entered the fray with their java.util.regex package"),

- **wherein the side-effects are executed as a step of the matching of a regular expression to data, the step occurring just after the "winning" match has been determined, and just before the next statement in code outside of the regular expression is executed** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading"),

- **wherein the side-effects do not alter the matching characteristics of a regular expression composition or sub-composition** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language"),

- wherein the side-effects are triggered if and only if the regex composition or sub-composition associated with its side-effects is ultimately matched to data; and wherein the method allows solutions to data matching problems to encode a greater portion of the solution within regular expressions and a lesser portion of the solution within surrounding code (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 59, the rejection of claim 58 is incorporated and further, Friedl discloses that **a regex composition grammar form (termed the "capture-pattern") allows the match to a regex sub-composition not only to be captured, but to be captured into any named variable (of type String) available in the scope in which its matchable Pattern sub-composition is defined** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to <u>describe and parse text</u>. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and <u>mutilate all kinds of text and data</u>", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for <u>operator overloading</u>").

As per claim 60, the rejection of claim 59 is incorporated and further, Friedl discloses that **the "capture-pattern" syntax further allows the capture reference to be any valid reference of type String, in which the reference expression's elements are available in the scope in which the capture-pattern's matchable Pattern sub- composition is defined** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to <u>describe and parse text</u>. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 61, the rejection of claim 58 is incorporated and further, Friedl discloses that **a regex composition grammar form (termed the "do-pattern") allows side-effect producing statements to be wrapped around a matchable Pattern sub-composition, wherein the "do-pattern" syntax consists of a suitable keyword signaling its start, such as do, followed by a pre-list of statements, followed by a Pattern sub- composition which determines what the "do-pattern" matches, followed by a post-list of statements, the pre-list and post-list statements comprising statements within the same programming language in which regular expressions are composed** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 62, the rejection of claim 61 is incorporated and further, Friedl discloses that **the pre-list and post-list statements of the "do-pattern" comprise**

any of the statement grammar forms of the programming language of the invention, except for the return statement and the tokenize statement (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 63, the rejection of claim 61 is incorporated and further, Friedl discloses that **the "do-pattern" establishes its own scope, such that temporary variables) are normally defined in the pre-list, and are usable (once defined) in the pre-list, and/or the matchable Pattern sub-composition, and/or the post-list** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 64, the rejection of claim 59 or 61 is incorporated and further, Friedl discloses that **"do-patterns" and "capture- patterns" involved in a match and/or sub-match associate side-effect instructions to the matching process, wherein the instructions are automatically executed if and only if the "do-patterns" and "capture-patterns" are involved in the "winning" match; wherein the instructions are a direct translation of pre-list and post-list statements in a "do-pattern"; and wherein the instructions are generated for a "capture-pattern" by modeling the "capture-pattern" as a low- level "do-pattern"** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 65, the rejection of claim 59 or 61 is incorporated and further, Friedl discloses that **"do-pattern" and/or "capture pattern" side-effects selected for execution along with the "winning" match are determined unambiguously based on a set of ambiguity resolution rules appropriate (in the presence of side-effects) to unions, concatenations, repeats, and iterates** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 66, the rejection of claim 58 is incorporated and further, Friedl discloses that **a binary regex composition, called the "subjunctive", has two forms, but and butnot, both of which consist of a primary term and a secondary term, wherein the primary term contributes not only a primary matching characteristic, but also side-effects, wherein the secondary term contributes no side-effects (because they are automatically inhibited), and wherein the secondary term is used to restrict the possible matches of the primary term** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 67, the rejection of claim 66 is incorporated and further, Friedl discloses that **a suitably named subjunctive keyword, such as but, signals that the secondary term restricts the matches of the primary to the extent that the secondary is able to match the same data matched by the primary at the same time, and wherein the secondary term in no way changes the side-effects implied by the primary's match (assuming the primary is allowed by the secondary to match)** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 68, the rejection of claim 66 is incorporated and further, Friedl discloses that **a suitably named subjunctive keyword, such as butnot, signals that the secondary term restricts the matches of the primary to the extent that the secondary is not able (in any way) to match the same data matched by the primary at the same time, and wherein the secondary term in no way changes the side-effects implied by the primary's match (assuming the primary is allowed by the secondary to match)** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 69, the rejection of claim 58 is incorporated and further, Friedl discloses that **regular expressions can be encapsulated as named, re-usable production rules, the rules having the characteristic of parameterizability, the rules comprising bodies which can incorporate side-effect producing regex compositions, such as “do-patterns" and “capture-patterns"** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data”, and p. 38:2-4, “Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience”, and C# and C++ provide for operator overloading").

As per claim 70, the rejection of claim 69 is incorporated and further, Friedl discloses that **the production rule grammar permits the parameteritzability of side-effects and/or matching characteristics** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 71, the rejection of claim 70 is incorporated and further, Friedl discloses that **"in" "out" and "in out" parameters of a production rule can be used to parameterize the side-effects of the rule, wherein such parameters are usable to formulate the capture-reference of a capture-pattern" (if such a "capture-pattern" is found within the rule's body) , and wherein such parameters are usable in the formulation of the pre-list and/or post-list statements of a "do-pattern" (if such a "do- pattern" is found within the rule's body)** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 72, the rejection of claim 70 is incorporated and further, Friedl discloses that **parameters of the "in" variety can be used to parameterize the matching characteristics of the rule** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 73, the rejection of claim 72 is incorporated and further, Friedl discloses that **a parameter of the "in" variety can be declared of type Pattern, and used not only to allow its rule to build upon the matching characteristics of the actual Pattern object being passed/bound, but also to allow its rule to inherit the side- effect characteristics of the actual Pattern parameter being passed/bound; wherein the parameter is deemed a target of the rule, consistent with the "find-first" and "find-nth" design patterns** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading).

As per claim 74, the rejection of claim 69 is incorporated and further, Friedl discloses that **the association of actual parameter values (or references to variables) to a signature compatible production rule results in an "instantiation" object that is type-compatible with the Pattern datatype, and can therefore be used anywhere a Pattern object can be used** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 75, the rejection of claim 58 is incorporated and further, Friedl discloses that **a single (suitably composed) production rule encapsulates the entire problem of directed capture of multi-dimensional data, leveraging <Edo-patterns" and "capture-patterns" and leverages the ability of production rule parameters to be used in the "do-pattern's" and "capture-patterns"** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 76, the rejection of claim 58 is incorporated and further, Friedl discloses that **production rules that have a base-case which must not match any data are defined based on the use of a conditional operator and the reject literal, wherein the reject literal serves as a base case** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 77, the rejection of claim 58 is incorporated and further, Friedl discloses that **an inhibit keyword transforms any Pattern object to a new Pattern object that inherits all of the original's matching characteristics and none of the original's side-effect characteristics** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 78, the rejection of claim 58 is incorporated and further, Friedl discloses a **computer system for interpreting or compiling and then executing a programming language with support for regular expressions, including a plurality of support for integrating side-effects into regular expressions, the system comprising: a computer processor; the programming language and regex grammar of claim 58; a computer program which executes computer program modules/scripts written in the programming language and, while the computer program is being hosted by the computer processor, the computer program is also the host for the computer program modules/scripts which are its target executables; a data input device providing a user-defined program module/script (written by user according to the grammatical rules of the programming language) to the host computer program; a data input device providing a source of data to be processed by the regular expressions of a program module/script; a set of virtual instructions, into which instructions the program modules/scripts are translated, and which instructions are executed by a subcomponent of the host computer program called the "virtual machine"' and a subset of virtual instructions, which map accessibility functions of input/output devices in the system to instructions, so that program modules/scripts can access input/output devices and thereby provide the programmer with output consistent with the user**

defined program module/script (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 79, the rejection of claim 78 is incorporated and further, Friedl discloses that **support (in the "host computer program") for regular expressions and support for integrating side-effects into regular expressions by: extending the virtual machine, including its instruction set, into which not only the normal statements and function calls of the programming language can be defined, but also its regular expression compositions, including also instructions that execute regular expressions against data; defining a set of regular expression composition instructions that allow every regular expression of the programming language to be modeled in the virtual machine as an immutable object; implementing a group of subset construction techniques that permit the transformation of every regular expression object in the virtual machine into a**

form suitable for execution against data; implementing a runtime automata execution technique that allows the execution of all of the regular expressions of the invention, once transformed into automata, against data, in such a way that each character from the stream is examined at most once; implementing the runtime automata execution technique in such a way that side- effects of the automata are accumulated as instructions to a "winning" thread, and executed if and only if the regular expressions to which they correspond are involved in a "winning" match against the data; implementing the compositions and subset constructions of regular expressions and their automata with inclusion of both instruction-arcs to model side- effects as well as arcnum-sequences to serve as a criteria for unambiguously selecting a "winning" match to data, in the presence of side-effects; and implementing the runtime automata execution technique in such a way that a set of "automata-threads" is maintained, that represent the set of all ways in which data not only can be matched, but also all ways in which side-effects can be accumulated by the ultimate "winning" thread (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 80, the rejection of claim 79 is incorporated and further, Friedl discloses that **side-effects are modeled as instruction-arcs, which are placed into composition graphs (automata) to fully model both "do-patterns" and "capture-patterns," wherein the instruction-arcs imply the existence of arcnum-sequences that must be attached to "eating" arcs of the automata in order to allow the runtime automata execution technique to make decisions on automata-thread preference** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 81, the rejection of claim 79 is incorporated and further, Friedl discloses **a subset construction technique that allows a modeling automata/graph to be converted into a form suitable for execution, including the presence of both**

instruction-arcs and arcnum-sequences (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and <u>mutilate all kinds of text and data</u>", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 82, the rejection of claim 79 is incorporated and further, Friedl discloses **a subset construction technique that allows a subjunctive convolution to be performed to represent a subjunctive expression, resulting in yet another composition graph that models semantics of the subjunctive expression by accurately representing both possible matches and possible side-effects of the subjunctive expression** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, regular expressions can add, remove, isolate, and generally fold, spindle, and <u>mutilate</u> all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

As per claim 83, the rejection of claim 79 is incorporated and further, Friedl discloses that **the runtime automata execution technique includes a pruning step to limit proliferation of the automata threads to be tracked, thus preventing an exponential explosion of automata threads in the presence of nested, repeating side-effect compositions** (p. 4:14-17, " Regular expressions are the key to powerful, flexible, and efficient text processing. Regular expressions themselves, with a general pattern notation almost like a mini programming language, allow you to describe and parse text. With additional support provided by the particular tool being used, <u>regular expressions</u> can add, remove, isolate, and generally fold, spindle, and mutilate all kinds of text and data", and p. 38:2-4, "Microsoft's .NET Framework, usable with Visual Basic, C#, and C++, offers a shared regular-expression library that unifies regex semantics among the languages. It's a full-featured, powerful engine that allows you the maximum flexibility in balancing speed and convenience", and C# and C++ provide for operator overloading").

Response to Arguments

7. *In the remarks, the applicant has argued substantially that:*

1) The cited art does not disclose the limitations of new claims 58-83, at p. 37:40-49:2.

Examiner's response:

1) In response to applicant's argument that the references fail to show the limitations of the new claims, it is noted that the newly added limitations upon which applicant relies are fully addressed in the above art rejection.

Conclusion

8. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Andre R. Fowlkes whose telephone number is (571) 272-3697. The examiner can normally be reached on Monday - Friday, 8:00am-4:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571)272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

ARF

[illegible]

TUAN DAM
SUPERVISORY PATENT EXAMINER